1C

УДК 519.7

А.В. Стёпкин
Славянский государственный педагогический университет, Украина
stepkin.andrey@rambler.ru

Распознавание конечных графов тремя агентами

В статье рассматривается проблема распознавания конечных графов тремя агентами. Два агента-исследователя передвигаются по графу, считывают, анализируют и изменяют метки элементов графа, передают информацию о своих передвижениях агенту-экспериментатору, который и распознает исследуемый граф. Предложен алгоритм временной сложности $O(n^3)$ и емкостной – $O(n^2)$, который распознает любой конечный неориентированный граф. При распознавании каждый агент использует две различные краски (всего три краски). Метод основан на методе обхода графа в глубину.

Введение

Проблема распознавания среды широко рассматривается в литературе в различных контекстах [1]. Например, в [2] данная проблема решается с помощью двух агентов. Один агент-исследователь передвигается по неизвестному графу, считывает, анализирует и изменяет метки на элементах графа, а также обменивается данными с агентом-экспериментатором, который, используя полученную информацию, и восстанавливает исследуемый граф.

Данная статья посвящена исследованию возможности и сложности решения нашей проблемы с помощью трёх одновременно работающих агентов. Два агента-исследователя (АИ) A и B одновременно передвигаются по неизвестной среде, заданной конечным графом [3], и обмениваются данными с агентом-экспериментатором (АЭ), который и производит восстановление графа, используя информацию, полученную от АИ, а также передает АИ данные, необходимые для их дальнейшей работы.

Целью данной работы является создание алгоритма одновременной работы трёх агентов, в котором два АИ, будучи размещены в произвольных, несовпадающих вершинах рассматриваемого графа G, окрашенных цветом w, через конечное число шагов обойдут этот граф, пошагово обмениваясь необходимыми данными с АЭ. Агент-экспериментатор, в свою очередь, восстановит граф H, изоморфный G, то есть распознает граф G, используя данные, полученные от АИ.

Стратегия решения задачи

В работе рассматриваются конечные, неориентированные графы без петель и кратных ребер. Все неопределяемые понятия общеизвестны, с ними можно ознакомиться, например, в [4-7].

Работа алгоритма, рассматриваемого в данной статье, основана на методе поиска в глубину. Принцип работы этого метода можно описать следующим образом: агенты идут «в глубину», пока это возможно, возвращаются назад, ищут другой путь с еще не посещенными вершинами и не пройденными ребрами. В случае обнаружения смежной

вершины окрашенной в «чужой» цвет, агент метит все перешейки из текущей вершины в «чужую» область и сообщает второму АИ, через АЭ, о необходимости распознавания помеченных перешейков. Пока второй АИ выполняет процедуру распознавания перешейков, первый АИ не может метить новые перешейки. В случае отсутствия других возможных вариантов перемещения, кроме как пометить новый перешеек, первый АИ останавливается до того момента, пока второй АИ не распознает все помеченные перешейки.

Выполняя обход графа, агенты-исследователи A и B образуют соответственно красный и желтый пути. Принцип построения пути каждым из агентов можно описать следующим образом:

– при движении в новую вершину красный (желтый) путь удлиняется, при движении назад по «своему» пути – укорачивается;

– при движении назад для распознавания обратного ребра или перешейков длина пути не изменяется;

– вершина, из которой возможен лишь возврат по «своему» пути, либо вовсе отсутствуют варианты перемещения, окрашивается в черный цвет. Алгоритм заканчивает работу, когда красный и желтый пути становятся пустыми, а все вершины черными.

При обходе графа G, агентами создается неявная нумерация посещенных вершин: при первом посещении вершины она окрашивается агентом A в красный цвет (в желтый цвет в случае агента B), и ей фактически ставится в соответствие номер, равный значению переменной $Сч_A$ ($Сч_B$ для агента B). Обратим внимание, что $Сч_A$ и $Сч_B$ принимают соответственно нечетные и четные значения. На основе нумерации и происходит восстановление графа G путем построения графа H, изоморфного G.

В работе АИ можно выделить 5 режимов функционирования:

1. Работая в *обычном режиме*, АИ продвигается вперед по белым вершинам, окрашивая эти вершины, соединяющие их ребра и дальние инциденторы в «свой» цвет. При отсутствии возможных путей перемещения АИ возвращается назад, окрашивая пройденные вершины, соединяющие их ребра и ближние инциденторы в черный цвет. АИ завершает работу тогда, когда его исходная вершина, вследствие отсутствия возможных путей перемещения, окрашивается в черный цвет.

2. Если при работе в обычном режиме было обнаружено обратное ребро, то АИ прекращает работу в этом режиме и переключается в *режим распознавания обратных ребер*. АИ переходит по обратному ребру, окрашивая его в черный цвет, и по «своему» пути возвращается в вершину, в которой был изменен режим работы АИ. Достигнув этой вершины, агент переключается в обычный режим работы.

3. Если при работе в обычном режиме АИ обнаружил перешеек, то при условии, что все ранее помеченные данным агентом перешейки были распознаны, АИ переключается в режим пометки перешейков. В этом режиме АИ переходит по перешейку в «чужую» область, окрашивая ребро и дальний инцидентор в «свой» цвет. Далее АИ возвращается по этому перешейку, ничего не окрашивая, и ищет другие перешейки из этой вершины. Пометив все перешейки из данной вершины в «чужую» область, АИ сообщает об этом АЭ, который в свою очередь дает команду второму АИ о необходимости распознавания помеченных перешейков. По завершению режима пометки перешейков АЭ содержит информацию о количестве помеченных перешейков.

4. Получив от АЭ команду о необходимости распознавания перешейков, АИ завершает работу в обычном режиме и переключается в *режим распознавания перешейков*. Если в этот момент агент работает в режиме распознавания обратных ребер, то

переключение в режим распознавания перешейков будет совершено по завершении работы агента в текущем режиме. В режиме распознавания перешейков АИ возвращается назад по «своему» пути до обнаружения ближайшей вершины, инцидентной помеченному перешейку. Под помеченным перешейком понимается ребро, у которого ближний инцидентор, ребро и дальняя вершина этого ребра окрашены в «чужой» цвет. Далее возможны два случая:

4.1. *Помечен один перешеек.* АИ переходит по перешейку в «чужую» область, окрашивая его в «свой» цвет, а дальний инцидентор – в черный. На следующем шаге АИ возвращается по этому перешейку в «свою» область, окрашивая дальний инцидентор и перешеек в черный цвет.

4.2. *Помечено не менее двух перешейков*. АИ переходит по первому найденному помеченному перешейку в «чужую» область, окрашивая его в «свой» цвет, а дальний инцидентор – в черный. На следующем шаге АИ возвращается по этому перешейку в «свою» область, окрашивая его в черный, а оба инцидентора – в «свой» цвет. Далее АИ движется назад по «своему» пути, пока не будет найден следующий помеченный перешеек.

Далее возможны два варианта:

4.2.1. *Следующий помеченный перешеек не последний*. АИ переходит по найденному перешейку, окрашивая его в «свой» цвет, а дальний инцидентор – в черный. На следующем шаге АИ возвращается по этому перешейку в «свою» область, окрашивая его и оба инцидентора в черный цвет. И снова возвращается назад по «своему» пути до следующего помеченного перешейка.

4.2.2. *Следующий помеченный перешеек последний*. АИ переходит по найденному перешейку, окрашивая его в «свой» цвет, а дальний инцидентор – в черный. На следующем шаге АИ возвращается по этому перешейку в «свою» область, окрашивая его в черный, а оба инцидентора – в «свой» цвет.

АИ переходит по последнему перешейку в «чужую» область, окрашивая инциденторы в черный цвет. На следующем шаге АИ переходит по первому распознанному перешейку в «свою» область, окрашивая пройденные инциденторы в черный цвет.

Далее АИ движется вперед по «своему» пути, пока не вернется в вершину, в которой он переключился в режим распознавания перешейков.

5. При одновременном попадании двух АИ в одну белую вершину, каждый АИ окрашивает вершину наполовину, и она становится красно-желтой. Агент B на следующем шаге отступает назад по своему пути и переходит в режим пометки перешейков (при этом ребро, по которому он вернулся, уже посчитано как первый перешеек, длина желтого пути уменьшена на одну вершину, а из списков ребер и вершин удалены ребро и вершина, записанные туда агентом B на предыдущем шаге). Агент A видит разноцветную вершину как «свою», но при распознавании окрашивает в черный цвет обе половинки.

Алгоритмы обхода и восстановления

Рассмотрим непосредственно алгоритмы работы агентов, реализующие описанную выше стратегию. Процесс распознавания состоит из двух принципиально разных типов алгоритмов: «Обход» и «Восстановление». Первый тип алгоритма описывает обход графа G агентами-исследователями с целью проведения элементарных экспериментов и передачи необходимой информации АЭ. Второй тип алгоритма представляет собой анализ результатов элементарных экспериментов, в результате которого будет построен граф H, изоморфный распознаваемому графу G.

Алгоритм работы агента A:

1. Агент A красит $(\mu(v) := r)$;
2. запрос AN;
3. *if* $AN \neq 1$ *then do*
4. запрос BN;
5. *if* $BN = 0$ *then* МЕТИМ_ПЕР_А(v);
6. *else* ВЫБОР_ХОДА_А(v);
7. *end do*;
8. *else* РАСП_ПЕР_А(v);

Все процедуры, которые не описаны ниже, представлены в [8].

РАСП_ПЕР_А(v):

1. $Z := K$;
2. *if* в окрестности $O(v)$ нет ребра, у которого $(\mu(v,u) = y)$ *then do*
3. агент A выполняет ОТСТУП_А(v);
4. *go to* 2 данной процедуры;
5. *end do*;
6. *else do*
7. *if* $(((K = Z) or (K = 1)) and (Z \neq 1))$ *then* агент A выполняет РАСП_АВВ(v);
8. *else* агент A выполняет процедуру РАСП_АВВb(v);
9. агент A запрашивает у АЭ значение переменной K;
10. *if* $K \neq 0$ *then go to* 2 данной процедуры;
11. *else* агент A выполняет процедуру ОБН_А(v);
12. *if* $Z \neq 1$ *then do*
13. *if* в $O(v)$ есть ребро, у которого $(\mu((v,u),v) = r) and (\mu(v,u) = b) and (\mu((v,u),u) = r)$ *then do*
14. агент A выполняет процедуру ВПЕРЕД_AR_N(v);
15. *go to* 13 данной процедуры;
16. *end do*;
17. *else if* в $O(v)$ есть ребро, у которого $(\mu(v,u) = r) and (\mu(u) = r) and (\mu((v,u),u) = r)$ *then do*
18. агент A выполняет процедуру ВПЕРЕД_ AR(v);
19. *go to* 17 данной процедуры;
20. *end do*;
21. *else go to* 2 алгоритма обхода;
22. *end do*;
23. *else go to* 17 данной процедуры;
24. *end do*;

Выполняя процедуру НАЗАД_А(v), агент A выбирает из окрестности $O(v)$ ребро, для которого выполняется условие $((\mu(v,u) = r) and (\mu((v,u),v) = r)) and (\mu(v) = r)$, и переходит по нему в вершину u. При этом, окрашивает $\mu(v) := b,\ \mu(v,u) := b, \mu((v,u),v) := b$, выполняет присвоение $v := u$ и записывает в список M сообщение: НАЗАД_А.

РАСП_А(v):

1. Агент A выбирает из окрестности $O(v)$ ребро (v,u), у которого $(\mu(v) = \mu(u) = r) and (\mu(v,u) = w)$ и переходит по нему в вершину u;
2. агент A красит $\mu(v,u) = b$;
3. агент A записывает в список M сообщение: ОБРАТНОЕ_РЕБРО_А;
4. *while* в $O(u)$ есть ребро (u,l), у которого $(\mu(u,l) = r) and (\mu((u,l),l) = r) and (\mu(l) = r)$ *do*

5. агент A переходит по ребру (u,l) в вершину l;
6. $u := l$;
7. агент A записывает в список M сообщение: ОТСТУПИЛ_А;
8. *end do*;
9. агент A записывает в список M сообщение: РЕБРО_РАСПОЗНАНО_А;

При выполнении процедуры РАСП_АВВ(v), агент A выбирает из окрестности $O(v)$ ребро (v,u), для которого выполняется условие $\mu(v,u) = y$, и переходит по нему в вершину u, производя окрашивание $\mu(v,u) := r$, $\mu((v,u),u) := b$. Выполняет присвоение $v := u$ и записывает в список M сообщение: ВПЕРЕД_АВВ. После чего агент A выбирает из окрестности $O(v)$ ребро (v,u), для которого выполняется условие $((\mu(v,u) = r) and (\mu((v,u),v) = b))$, и переходит по нему в вершину u, окрашивая $\mu((v,u),v) := r, \mu(v,u) := b, \mu((v,u),u) := r$, выполняет присвоение $v := u$ и записывает в список M сообщение: НАЗАД_АВВ.

Процедура РАСП_АВВb(v) аналогична процедуре РАСП_АВВ(v), с отличием в том, что, выполняя возврат по перешейку в «свою» область, агент A окрашивает ребро и дальний инцидентор следующим образом $\mu(v,u) := b$, $\mu((v,u),u) := b$.

Выполняя процедуру ВПЕРЕД_AR_N(v), агент A выбирает из окрестности $O(v)$ ребро (v,u), удовлетворяющее условию $(\mu((v,u),v) = r) and (\mu(v,u) = b) and (\mu((v,u),u) = r)$, переходит по нему в вершину u, окрашивая $\mu((v,u),v) := b$, $\mu((v,u),u) := b$. После чего выполняет присвоение $v := u$ и записывает в список M сообщение: ВПЕРЕД_AR_N.

Выполняя процедуру СТОП_А(v), агент A красит $\mu(v) := b$ и завершает работу.

Алгоритм работы агента B:

1. Агент B красит $(\mu(s) := y)$;
2. запрос BN;
3. *if* $BN \neq 1$ *then do*
4. запрос AN;
5. *if* $\mu(s) = ry$ *then do*
6. агент B выполняет процедуру ВОЗВРАТ_В(s);
7. агент B выполняет процедуру МЕТИМ_ПЕР_В(s);
8. *end do*;
9. *else if* $AN = 0$ *then* МЕТИМ_ПЕР_В(s);
10. *else* ВЫБОР_ХОДА_В(s);
11. *end do*;
12. *else* РАСП_ПЕР_В(s);

Все процедуры агента B, которые не рассмотрены ниже, аналогичны процедурам агента A.

При выполнении процедуры МЕТИМ_ПЕР_В(s), агент B проверяет наличие в окрестности $O(s)$ ребра (s,z), у которого $(\mu(s,z) = w) and ((\mu(z) = r) or (\mu(z) = ry))$. Если такое ребро обнаружено и в вершине z находится агент A, то агент B выполняет процедуру СТОИТ_ В(s) и возвращается в начало данной процедуры. Если же в вершине z нет агента A, то агент B выполняет процедуру МЕТИМ_ВА(s) и возвращается в начало данной процедуры.

Если в окрестности $O(s)$ не обнаружено ребра, которое удовлетворяет условию $(\mu(s,z)=w)and((\mu(z)=r)or(\mu(z)=ry))$, то агент B запрашивает значение переменной L. При этом если $L=0$, то агент B выполняет процедуру ВЫБОР_ХОДА_В(s), иначе агент B выполняет процедуру ФИКС_В(s) и возвращается в строку 2 АО.

ВЫБОР_ХОДА_В(s):

if в $O(s)$ обнаружено ребро, у которого $(\mu(s,z)=w)and(\mu(z)=\mu(s)=y)$ *then do*
    агент B выполняет процедуру РАСП_В(s);
    go to 2 алгоритма обхода;
    end do;
else if в $O(s)$ обнаружено ребро, у которого $(\mu(s,z)=w)and(\mu(z)=w)$ *then do*
    агент B выполняет процедуру ВПЕРЕД_В(s);
    go to 2 алгоритма обхода;
    end do;
else if в $O(s)$ есть ребро, у которого $(\mu(s,z)=w)and((\mu(z)=r)or(\mu(z)=ry))$ *then do*
    агент B выполняет процедуру СТОИТ_В(s);
    go to 2 алгоритма обхода;
    end do;
else if в $O(s)$ есть ребро, у которого $(\mu(s,z)=r)$ *then do*
    агент B выполняет процедуру СТОИТ_В(s);
    go to 2 алгоритма обхода;
    end do;
else if в $O(s)$ есть ребро, у которого $(\mu(s,z)=y)and((\mu(z)=r)or(\mu(z)=ry))$ *then do*
    агент B выполняет процедуру СТОИТ_В(s);
    go to 4 алгоритма обхода;
    end do;
else if в $O(s)$ есть ребро, у которого $(\mu(s,z)=y)and(\mu(s)=y)and(\mu((s,z),s)=y)$ *then do*
    агент B выполняет процедуру НАЗАД_В(s);
    go to 2 алгоритма обхода;
    end do;
else агент B выполняет процедуру СТОП_В;

При выполнении процедуры МЕТИМ_ВА(s), агент B выбирает из окрестности $O(s)$ произвольное ребро (s,z), для которого выполняется условие $((\mu(s,z)=w)and((\mu(z)==r)or(\mu(z)=ry)))$, переходит по нему в вершину z, окрашивая $\mu(s,z):=y,\ \mu((s,z),z):=y$, выполняет присвоение $s:=z$ и записывает в список N сообщение: ВПЕРЕД_ВА. Далее агент B выбирает из окрестности $O(s)$ ребро (s,z), у которого $((\mu(s,z)=y)and((\mu(s)==ry)))$, переходит по нему в вершину z, выполняет присвоение $s:=z$ и записывает в список N сообщение: НАЗАД_ВА.

Выполняя процедуру ВОЗВРАТ_В(s), агент B выбирает из окрестности $O(s)$ ребро (s,z), у которого $(\mu(s,z)=y)and(\mu((s,z),s)=y)$, переходит по нему в вершину z, выполняет присвоение $s:=z$ и записывает в список N сообщение: ВОЗВРАТ_В.

Алгоритм «Восстановление» и процедуры, которые не рассмотрены ниже, изложены в [8] с поправкой, что, при использовании цикла с предусловием, условие имеет вид: $(M\neq\varnothing)or(N\neq\varnothing)$.

ОБР_СП_А():

1. *if* $Mes =$ "*ВПЕРЕД_А*" *then* ВПЕРЕД_А();
2. *if* $Mes =$ "*ВПЕРЕД_АВ*" *then* ВПЕРЕД_АВ();
3. *if* $Mes =$ "*ВПЕРЕД_АВВ*" *then* ВПЕРЕД_АВВ();
4. *if* $Mes =$ "*НАЗАД_А*" *then* НАЗАД_А();
5. *if* $Mes =$ "*НАЗАД_АВ*" *then* НАЗАД_АВ();
6. *if* $Mes =$ "*НАЗАД_АВВ*" *then* НАЗАД_АВВ();
7. *if* $Mes =$ "*ФИКС_А*" *then* ФИКС_А();
8. *if* $Mes =$ "*ОБН_А*" *then* ОБН_А();
9. *if* $Mes =$ "*ОТСТУПИЛ_А*" *then* ОТСТУПИЛ_А();
10. *if* $Mes =$ "*РЕБРО_РАСПОЗНАНО_А*" *then* РЕБРО_РАСПОЗНАНО_А();
11. *if* $Mes =$ "*ОТСТУП_А*" *then* ОТСТУП_А();

ОТСТУП_А(): $i := i + 1$;

ВПЕРЕД_АВВ(): $E_H := E_H \cup \{(N_B, r(t-i))\}$;

ОТСТУПИЛ_А(): $i := i + 1$;

РЕБРО_РАСПОЗНАНО_А(): $E_H := E_H \cup \{(r(t), r(t-i))\}$; $i := 0$;

Процедуры работы со списком команд от агента B, которые не рассмотрены ниже, аналогичны процедурам работы со списком команд от агента A.

ОБР_СП_В():

1. *if* $Mes =$ "*ВПЕРЕД_В*" *then* ВПЕРЕД_В();
2. *if* $Mes =$ "*ВПЕРЕД_ВА*" *then* ВПЕРЕД_ВА();
3. *if* $Mes =$ "*ВПЕРЕД_ВАА*" *then* ВПЕРЕД_ВАА();
4. *if* $Mes =$ "*НАЗАД_В*" *then* НАЗАД_В ();
5. *if* $Mes =$ "*НАЗАД_ВА*" *then* НАЗАД_ВА();
6. *if* $Mes =$ "*НАЗАД_ВАА*" *then* НАЗАД_ВАА();
7. *if* $Mes =$ "*ФИКС_В*" *then* ФИКС_В();
8. *if* $Mes =$ "*ОБН_В*" *then* ОБН_В();
9. *if* $Mes =$ "*ОТСТУПИЛ_В* " *then* ОТСТУПИЛ_В();
10. *if* $Mes =$ "*РЕБРО_РАСПОЗНАНО_В*" *then* РЕБРО_РАСПОЗНАНО_В();
11. *if* $Mes =$ "*ОТСТУП_В*" *then* ОТСТУП_В();
12. *if* $Mes =$ "*ВОЗВРАТ_В*" *then* ВОЗВРАТ_В().

ВОЗВРАТ_В(): $E_H := E_H \setminus \{(y(p-1), y(p))\}$; $V_H := V_H \setminus \{Сч_B\}$; $Сч_B := Сч_B - 2$; $p := p - 1$; $y(p) := Сч_B$; $L := 1$; $K := K + 1$.

Свойства алгоритма распознавания

В начале алгоритма, при $n \geq 3$, как минимум, по одному разу выполняются процедуры: ВПЕРЕД_А(v), ВПЕРЕД_А() и ВПЕРЕД_В(s), ВПЕРЕД_В(). Выполняя процедуры ВПЕРЕД_А(v) и ВПЕРЕД_В(s), АИ посещают новые вершины исследуемого графа G. Процедурами агента АЭ ВПЕРЕД_А() и ВПЕРЕД_В() создаются две новые вершины (по одной вершине для каждой из процедур) графа H.

При одновременном попадании двух АИ в одну белую вершину процедурами ВПЕРЕД_А() и ВПЕРЕД_В() будет создано две новые вершины графа H. Одна из этих

двух вершин (вершина, созданная агентом B) будет удалена командой ВОЗВРАТ_В(), так как она дублирует вершину, созданную агентом A. Таким образом, процесс выполнения описанного алгоритма индуцирует отображение $\varphi : V_G \to V_H$ вершин графа G в вершины графа H. Причем $\varphi(v) = t$ (когда вершина v окрашена в красный цвет) и $t = Сч_A$) и $\varphi(s) = p$ (когда вершина s окрашена в желтый цвет и $p = Сч_B$). Указанное отображение естественным образом устанавливает неявную нумерацию вершин графа G. Более того, отображение φ является биекцией, поскольку в связном графе G все вершины достижимы из начальных вершин. Поэтому все вершины посещаются агентами, то есть окрашиваются в красный и желтый цвета.

Из описания алгоритма следует, что АИ проходят все ребра графа G, поскольку при окончании алгоритма все ребра становятся черными. При выполнении процедуры ВПЕРЕД_А() или ВПЕРЕД_В() АЭ распознает древесное ребро (v,u) и так нумерует вершину u, что ребру (v,u) однозначно соответствует ребро $(\varphi(v),\varphi(u))$ графа H. При выполнении процедур РЕБРО_РАСПОЗНАНО_А() или РЕБРО_РАСПОЗНАНО_В() АЭ распознает обратное ребро (v,u) графа G и ставит ему в однозначное соответствие ребро $(\varphi(v),\varphi(u))$ графа H. При выполнении процедур ФИКС_А(), ВПЕРЕД_АВВ() или ФИКС_В(), ВПЕРЕД_ВАА() АЭ распознает перешеек (v,u) графа G и ставит ему в однозначное соответствие ребро $(\varphi(v),\varphi(u))$ графа H. Следовательно, φ является изоморфизмом графа G на граф H.

Теорема 1. Выполняя алгоритм распознавания, агенты распознают любой граф G с точностью до изоморфизма.

Подсчитаем временную и емкостную сложность в равномерной шкале [5]. Для этого рассмотрим свойства красного и желтого путей. Из описания алгоритма следует, что на каждом шаге алгоритма красный (желтый) путь – это простой путь, соединяющий начальную вершину v (s - в случае агента B) с номером $\varphi(v) = 1$ $(\varphi(s) = 2)$ с вершиной u (z) с номером $\varphi(u) = Сч_A$ $(\varphi(z) = Сч_B)$. Следовательно, общая длина красного и желтого пути не превосходит n.

При выполнении процедур ВПЕРЕД_А(v), ВПЕРЕД_В(s) и НАЗАД_А(v), НАЗАД_В(s) АИ проходят одно ребро. При выполнении процедур РАСП_А(v), РАСП_В(s) АИ проходят одно обратное ребро и не более $n-2$ (изначально одна вершина уже окрашена в «чужой» цвет) ребер красного и желтого путей. При выполнении процедур РАСП_А(v), РАСП_В(s) АИ проходят фактически цикл, состоящий из обратного ребра и некоторого конечного отрезка красного (желтого) пути, соединяющего вершины, инцидентные обратному ребру. При выполнении процедур МЕТИМ_АВ(v), МЕТИМ_ВА(s) и РАСП_АВВ(v), РАСП_АВВb(v), РАСП_ВАА(s), РАСП_ВААb(s) оба АИ проходят один и тот же перешеек, сначала в одном направлении, потом в обратном. Выполняя процедуры ВПЕРЕД_AR(v), ВПЕРЕД_BR(s) и ОТСТУП_А(v), ОТСТУП_В(s), АИ проходят одно красное (желтое) ребро. При выполнении процедур ВПЕРЕД_AR_N(v), ВПЕРЕД_BR_N(s) АИ проходят одно черное ребро. При выполнении процедур ФИКС_А(v), ФИКС_В(s) и ОБН_А(v), ОБН_В(s) АИ не передвигаются, а только делают записи в свой список команд для АЭ, на что так же уходит один ход.

При подсчете временной сложности алгоритма будем считать, что инициализация алгоритма, анализ окрестности $O(v)$ рабочей вершины и выбор одной из возможных процедур занимают некоторое постоянное число единиц времени. Также будем считать,

что выбор ребер, проход по ним АИ и обработка команд АЭ, полученных на данном этапе от АИ, осуществляется за 1 единицу времени. Тогда временная сложность алгоритма определяется следующими соотношениями:

1. Процедуры ВПЕРЕД_А(v), ВПЕРЕД_В(s), НАЗАД_А(v) и НАЗАД_В(s) выполняются не более чем $2\times(n-1)$ раз, общее время их выполнения оценивается как $O(n)$.

2. На выполнение процедур МЕТИМ_АВ(v), МЕТИМ_ВА(s), РАСП_АВВ(v), РАСП_АВВb(v), РАСП_ВАА(s) и РАСП_ВААb(s) уходит время, которое оценивается как $4\times O(n)\times n$, то есть как $O(n^2)$.

3. Каждая из пар процедур ВПЕРЕД_АR(v), ВПЕРЕД_ВR(s) и ОТСТУП_А(v), ОТСТУП_В(s) выполняются за время, оцениваемое как $O(n)\times n$, то есть как $O(n^2)$.

4. На выполнение процедур ВПЕРЕД_АR_N(v), ВПЕРЕД_ВR_N(s), ФИКС_А(v), ФИКС_В(s), ОБН_А(v) и ОБН_В(s) уходит время, оцениваемое как $3\times O(n)$, т.е. как $O(n)$.

5. Время, затрачиваемое на выполнение процедур РАСП_А(v) и РАСП_В(s), оценивается как $O(n)\times m$, то есть как $O(n^3)$.

6. Время выполнения процедур СТОИТ_А(v) и СТОИТ_В(s) в общей сложности для всех четырёх возможных случаев оценивается как $O(n)+O(n^2)=O(n^2)$.

Следовательно, суммарная временная сложность $T(n)$ алгоритма удовлетворяет соотношению: $T(n)=O(n^3)$.

Емкостная сложность $S(n)$ алгоритма определяется сложностью списков V_H, E_H, $r(1)...r(t), y(1)...y(p)$, сложность которых соответственно определяется величинами $O(n), O(n^2), O(n), O(n)$. Следовательно, $S(n)=O(n^2)$.

Теорема 2. Временная сложность алгоритма распознавания равна $O(n^3)$, а емкостная – $O(n^2)$. При этом алгоритм использует 3 краски.

Выводы

Основными результатами исследования являются: создание алгоритма работы трёх агентов, при условии, что АИ передвигаются по графу одновременно, а также решение проблемы окраски вершины, которая возникала при одновременном попадании двух АИ в одну белую вершину.

Предложен алгоритм точного распознавания графа среды временной сложности $O(n^3)$ и емкостной – $O(n^2)$. АИ имеют память, ограниченную числом n, и используют по две краски каждый (всего три краски).

На основе полученного алгоритма автор надеется создать новые более эффективные алгоритмы, которые позволят улучшить результаты, полученные в [4].

Литература

1. Albers S. Exploring unknown environments / S. Albers and M.R. Henzinger // SIAM Journal on Computing. – 2000. – № 29 (4). – P. 1164-1188.
2. Грунский И.С. Распознавание конечного графа блуждающим по нему агентом / И.С. Грунский, Е.А. Татаринов // Вестник Донецкого университета. Серия А. Естественные науки. – 2009. – Вып. 1. – С. 492-497.
3. Кудрявцев В.Б. Введение в теорию автоматов / Кудрявцев В.Б., Алешин С.В., Подкозлин А.С. – М. : Наука, 1985. – 320 с.

4. Грунский И.С. Распознавание конечного графа коллективом агентов / И.С. Грунский, А.В. Стёпкин // Труды ИПММ НАН Украины. – 2009. – Т. 19. – С. 43-52.
5. Ахо А. Построение и анализ вычислительных алгоритмов / Ахо А., Хопкрофт Дж., Ульман Дж. – М. : Мир, 1979. – 536 с.
6. Кормен Т. Алгоритмы: построение и анализ / Кормен Т., Лейзерсон Ч., Ривест Р. – М. : МЦНМО, 2001. – 960 с.
7. Касьянов В.Н. Графы в программировании: обработка, визуализация и применение / В.Н. Касьянов, В.А. Евстигнеев. – СПб. : БХВ – Петербург, 2003. – 1104 с.
8. Стёпкин А.В. Распознавание конечного графа коллективом агентов / А.В. Стёпкин // Інформатика та комп'ютерні технології-2009 : матеріали V міжнар. наук.-техн. конф. студентів, аспірантів та молодих вчених (Донецьк, 24 – 26 лист. 2009 р.). – Донецьк, 2009. – Т. 2. – С. 126-131.

Literatura

1. Albers S. SIAM Journal on Computing. № 29 (4). 2000 P. 1164-1188.
2. Grunsky I.S. Newsletter of Donetsk University. Series A. Natural Sciences.2009.Vol.1. P. 492-497.
3. Kudryavtsev V.B. Moscow: Nauka. 1985. 320 p.
4. Grunsky I.S. Studies of IAMM NASU. 2009. Vol.19. P. 43-52.
5. Aho A. Moscow: Mir. 1979. 536 p.
6. Kormen T. Moscow: MCCME. 2001. 960 p.
7. Kasyanov V.N. S-Ptb.: BHV. 2003. 1104 p.
8. Stepkin A.V. Informatics and computer tehnologies.2009. Materials of the V-th science conference for postgraduates and young scientists. Donetsk. 2009. Vol. 2. P. 126-131.

А.В. Стьопкін

Розпізнавання скінченних графів трьома агентами

У статті розглядається проблема розпізнавання скінченних графів трьома агентами. Два агенти-дослідники рухаються графом, зчитують, аналізують та змінюють помітки елементів графа, передають інформацію про свої переміщення агенту-експериментатору, який розпізнає досліджуваний граф. Запропоновано алгоритм часової складності $O(n^3)$ та ємнісної – $O(n^2)$, який розпізнає будь-який скінченний неорієнтований граф. Для розпізнавання кожному агенту необхідно дві різні фарби (усього три фарби). Метод базується на методі обходу графа в глибину.

A.V. Stepkin

Finite Graphs Exploration by Three Agents

The Problem of finite graphs exploration by three agents is considered in this work. Two agents-researchers move on graph, they read, analyze and change marks of graph elements, transfer the information about their movements and colorings to the agent-experimenter. It builds explored graph representation. The algorithm with $O(n^3)$ time (n is amount of nodes of graph) and $O(n^2)$ space complexities is proposed. It recognizes any finite non-oriented graph. For graph exploration each agent needs two different marks (three colors in total). The method is based on the depth-first traversal method.

Статья поступила в редакцию 16.05.2011.